Noname manuscript No.
(will be inserted by the editor)

Nonlocal discrete ∞-Poisson and Hamilton Jacobi equations

from stochastic game to generalized distances on images, meshes, and point clouds

Matthieu Toutain · Abderrahim Elmoataz · François Lozes · Amin Mansouri

Received: date / Accepted: date

Abstract In this paper we propose an adaptation of the ∞-Poisson equation on weighted graphs, and propose a finer expression of the ∞-Laplace operator with gradient terms on weighted graphs, by making the link with the biased version of the tug-of-war game. By using this formulation, we propose a hybrid ∞-Poisson Hamilton-Jacobi equation, and we show the link between this version of the ∞-Poisson equation and the adaptation of the eikonal equation on weighted graphs. Our motivation is to use this extension to compute distances on any discrete data that can be represented as a weighted graph. Through experiments and illustrations, we show that this formulation can be used in the resolution of many applications in image, 3D point clouds, and high dimensional data processing using a single framework.

Keywords Generalized distance · ∞-Poisson equation · Hamilton-Jacobi equation · Weighted graphs · Partial difference equations · Tug-of-war game

1 Introduction

The main goal of this paper is to adapt and to solve ∞-Poisson and Hamilton-Jacobi equation on general discrete domain: a weighted graph of arbitrary topology. This adaptation is introduced as Partial difference Equations (PdEs) which are convex combinations of discrete ∞-Laplacian and discrete upwind gradient on graphs. These PdEs can be also interpreted as a simple combination of two discrete upwind gradients.

Our motivation is to give a simple and unified numerical scheme to approximate generalized distances on images, meshes, point clouds, or any data that can be represented as a weighted graph.

1.1 Introduction and motivations

Computing distance function has many applications in numerous area including image processing, computer graphics, robotics, or computational geometry. In addition, having the distance functions from a seed to a target, one can compute the corresponding geodesic path, which is used in many applications, to compute skeletons, voronoi diagrams, or to perform mesh editing.

Several distance functions approximation methods are based on partial differential equations (PDEs), in particular the eikonal equation, Poisson, ∞-Poisson, or screened Poisson [11,17,7,13,24].

In the context of a regular grid, solving numerically these equations is straightforward. The regularity of the grid provides a domain that is amendable, in the case of the Poisson equation, one can use several methods to compute a solution, such as using the Fourier transform, or a multigrid technique [11]. For ∞-Poisson equation or eikonal equation, one can use finite differences [18], or finite element methods. For meshes or general curved surfaces, solving these equations is more challenging. The numerical treatment of these PDEs requires a suitable representation of the geometry of the surface. One can use the parameterization of the surface as triangu-

M. Toutain
University of Caen, Lower Normandy, France
E-mail: matthieu.toutain@unicaen.fr

A. Elmoataz
University of Caen, Lower Normandy, France

F. Lozes
University of Caen, Lower Normandy, France

A. Mansouri
Le2i Laboratory, University of Bourgogne, France

lated meshes and uses either explicit representations to define differential operators on it, or the intrinsic geometry to define differential operators directly on the triangles. Another way is to represent the surface implicitly by level sets or using the closest point method [23]. Moreover, in the case of 3d point clouds, the connectivity of the points is not provided, which adds additional problems in the processing of these data.

In this paper, we propose a simple and unified numerical method for solving and adaptation of the ∞-Poisson and Hamilton-Jacobi equation on both regular or nonregular discrete domains. Using the framework of PdEs [9,10], first we interpret the tug-of-war game related to ∞-Laplacian and Hamilton Jacobi equations as PdEs on particular Euclidean graphs. Then, extending these same equations on weighted graphs, we give a general PdE which coefficients are data dependent. Setting differently these coefficients, we can adapt the general scheme to different applications on images, meshes, point clouds, and so on. The main contributions of this article are the following:

We propose an interpretation of both continuous ∞-Poisson and a hybrid ∞-Poisson-Hamilton-Jacobi equation, as PdEs on particular graphs. We propose an extension of these PdEs on weighted graphs of arbitrary topology, and show that these general PdEs are related to nonlocal tug-of-war game. We also show the connection with nonlocal continuous PDEs.

We propose to solve these PdEs with a simple morphological scheme: nonlocal erosion and dilatation operator type.

1.2 Tug-of-war game and ∞-Laplacian type equation

Let $\Omega \subset \mathbb{R}^n$, and $\partial\Omega$ its boundaries. We denote by $d(x)$ the minimal distance from $x \in \Omega$ to $\partial\Omega$. A common approach for simple approximation of a smooth distance function consists of solving the Dirichlet problem for the Poisson equation, which is defined as:

$$\begin{cases} \Delta u(x) = h(x) & , x \in \Omega, \\ u(x) = g(x) & , x \in \partial\Omega, \end{cases} \tag{1}$$

where u, h, and g are real-valued functions on the domain Ω, and Δ is the Laplace operator. This equation is classically used in various fields, such as electrostatics, Newtonian gravity, and more recently in surface reconstruction [15,4]. To compute an estimation of the distance from a point $x \in \Omega$ to $\partial\Omega$, a common approach is to set $h(x) = -1$ and $g(x) = 0$. In image processing, it was used *e.g.* to represent shapes [11]. In this case, the domain is a two dimensional grid, and solving this equation can be interpreted as the mean time a random walker would hit the boundary $\partial\Omega$ of Ω, starting from a point x in Ω.

One can also consider the p-Poisson equation, which is a natural generalization of Eq.(1):

$$\begin{cases} \Delta_p u(x) = -1 & , x \in \Omega, \\ u(x) = 0 & , x \in \partial\Omega, \end{cases} \tag{2}$$

where $\Delta_p u(x) = \operatorname{div}(|\nabla u(x)|^{p-2}\nabla u(x))$ is the p-Laplacian. One can see that when using $p = 2$, we recover the Poisson equation. As $p \to \infty$, it can be shown that $u(x) \to d(x)$ [14], giving the following equation:

$$\begin{cases} \Delta_\infty u(x) = -1 & , x \in \Omega, \\ u(x) = 0 & , x \in \partial\Omega, \end{cases} \tag{3}$$

where $\Delta_\infty u = \sum_{i,j} \frac{\partial^2 u}{\partial x_i \partial x_j} \frac{\partial u}{\partial x_i} \frac{\partial u}{\partial x_j}$ is ∞-Laplacian.

In a recent paper Peres et al. [21] have shown that the ∞-Poisson is connected to a stochastic game called tug-of-war. Let us briefly review the notion of tug-of-war game. Let $\Omega \subset \mathbb{R}^n$ be a Euclidean space, $h : \Omega \to \mathbb{R}$ the running payoff function, and $g : \partial\Omega \to \mathbb{R}$ the payoff function. Fix a number $\varepsilon > 0$. The dynamics of the game are as follows. A token is placed at an initial position $x_0 \in \Omega$. At the kth stage of the game, Player I and Player II select points x_k^I and x_k^{II}, respectively, each belonging to a specified set $B_\varepsilon(x_{k-1}) \subseteq \Omega$ (where $B_\varepsilon(x_{k-1})$ is the ε-ball centered in x_{k-1}. The game token is then moved to x_k, where x_k can be either x_k^I or x_k^{II} with probability $P = \frac{1}{2}$. In other words, a fair coin is tossed to determine where the token is placed. After the kth stage of the game, if $x_k \in \Omega$ then the game continue to stage $k+1$. Otherwise, if $x_k \in \partial\Omega$, the game ends and Player II pays Player I the amount $g(x_k) + \varepsilon^2 \sum_{j=0}^{k-1} h(x_j)$. Player I attempts to maximize the payoff while Player II attempts to minimize it. If both player are using optimal strategy, according to the dynamics programming principle, the value functions for Player I and Player II for standard ε-turn tug-of-war satisfy the relation

$$\begin{cases} u^\varepsilon(x) = \frac{1}{2}\left[\sup\limits_{y \in B_\varepsilon(x)} u^\varepsilon(y) + \inf\limits_{y \in B_\varepsilon(x)} u^\varepsilon(y)\right] + \varepsilon^2 h(x), \\ x \in \Omega, \\ u(x) = g(x), x \in \partial\Omega. \end{cases} \tag{4}$$

The authors of [21] have shown that when $h = 0$, or $\min h > 0$ or $\max h < 0$, the value function u^ε converges to the solution of the normalized ∞-Poisson equation:

$$\begin{cases} -\Delta_\infty^N u(x) = h(x) & , x \in \Omega, \\ u(x) = g(x) & , x \in \partial\Omega, \end{cases} \tag{5}$$

where $\Delta_\infty^N u = \frac{1}{|\nabla u|}\Delta_\infty$ is the normalized ∞-Laplacian.

If the game is modified as follows: we consider two fixed real number $\alpha > 0$, $\beta > 0$ and $\alpha + \beta = 1$. We can add bias in the tug-of-war game by using the same dynamics, but setting the probability to choose x_k^I as α and x_k^{II} as β. When the game is optimal, according to dynamic principle, the value function is:

$$\begin{cases} u^\varepsilon(x) = \alpha \sup\limits_{y\in B_\varepsilon(x)} u^\varepsilon(y) + \beta \inf\limits_{y\in B_\varepsilon(x)} u^\varepsilon(y) + \varepsilon^2 h(x), \\ x \in \Omega, \\ u(x) = g(x), x \in \partial\Omega. \end{cases} \tag{6}$$

This value function is related to the ∞-Laplacian with gradient terms: $-\Delta_\infty u(x) + c|\nabla u| = 1$, where c depends on the α and β values. This type of PDE and related stochastic game was studied in [20].

Paper organisation The rest of this paper is organized as Follows. In Section 2, we provide definitions, notation, and operators on graphs used in this work. In Section 3, we rewrite the value function of the tug-of-war and biased tug-of-war game in the context of our PdE framework, and we show the link with local and nonlocal PDEs. We also propose an iterative scheme to solve the proposed formulation on weighted graphs, and a morphological interpretation of this scheme. Finally, Section 4 presents experiments using our proposed formulation, to compute generalized distances and perform image segmentation and data clustering on discrete data, such as images, 3D point cloud, and unorganized high dimensional data.

2 Operators on graphs

As the core structure of our approach, in this section we provide notations and basics on weighted graphs, recall our formulations of difference, morphological differences, and gradients on weighted graphs. This section is a required preliminary to fully understand the different operators defined on weighted graphs that will be introduced in the following sections.

2.1 Basic notation

A *weighted graph* $G = (V, E, w)$ consists of a finite set V of $N \in \mathbb{N}$ vertices, a finite set $E \subseteq V \times V$ of edges, and a weight function $w : V \times V \to [0, 1]$. In our case the weight function represents a similarity measure between two vertices of the graph. We denote by $(u, v) \in E$ the edge that connects the vertices u and v and we write $u \sim v$ to denote two adjacent vertices. The *neighborhood* of a vertex u (i.e. the set of vertices adjacent to u) is denoted $N(u)$ and the *degree* of a vertex u is defined as $\delta_w(u) = \sum_{v\sim u} w(u, v)$.

Let $\mathcal{H}(V)$ be the Hilbert space of real valued functions on the vertices of the graph, i.e., each function $f : V \to \mathbb{R}$ in $\mathcal{H}(V)$ assigns a real value $f(u)$ to each vertex $u \in V$. For a function $f \in \mathcal{H}(V)$ the $\mathcal{L}^p(V)$ norm of f is given by:

$$\begin{aligned} \|f\|_p &= \Big(\sum_{u\in V} |f(u)|^p\Big)^{1/p}, \quad \text{for } 1 \leqslant p < \infty, \\ \|f\|_\infty &= \max_{u\in V}\big(|f(u)|\big), \quad \text{for } p = \infty. \end{aligned} \tag{7}$$

The Hilbert space $\mathcal{H}(V)$ is endowed with the following inner product: $\langle f, g\rangle_{\mathcal{H}(V)} = \sum_{u\in V} f(u)g(u)$ with $f, g \in \mathcal{H}(V)$. Similarly, let $\mathcal{H}(E)$ be the Hilbert space of real valued functions defined on the edges of the graph, i.e., each function $F : E \to \mathbb{R}$ in $\mathcal{H}(E)$ assigns a real value $F(u, v)$ to each edge $(u, v) \in E$. The Hilbert space $\mathcal{H}(E)$ is endowed with the following inner product: $\langle F, G\rangle_{\mathcal{H}(E)} = \sum_{u\in V}\sum_{v\in V} F(u, v)G(u, v)$ for $F, G \in \mathcal{H}(E)$.

Let $\mathcal{A} \subset V$ be a set of connected vertices, i.e., for all $u \in \mathcal{A}$ there exists a vertex $v \in \mathcal{A}$ with $(u, v) \in E$. We denote by $\partial\mathcal{A}$ the *(outer) boundary set* of $\mathcal{A}$, which is given by:

$$\partial\mathcal{A} = \{u \in \mathcal{A}^c : \exists v \in \mathcal{A} \text{ with } (u, v) \in E\}, \tag{8}$$

where $\mathcal{A}^c = V \setminus \mathcal{A}$ is the complementary set of $\mathcal{A}$ in V.

2.2 Nonlocal finite differences

Based on these basic notations, we are able to introduce the needed framework to translate differential operators and PDEs from the continuous setting to graphs. In particular the fundamental elements for this translation are nonlocal finite differences on graphs. For more detailed information on these operators we refer to [9, 2, 26]. In the following we assume that the considered graphs are connected, undirected, with neither self-loops nor multiple edges between vertices.

Let $G = (V, E, w)$ be a weighted graph and let $f \in \mathcal{H}(V)$ be a function on the set of vertices V of G. Then we can define the *weighted (nonlocal) finite difference* of f at a vertex $u \in V$ in direction of a vertex $v \in V$ as:

$$\partial_v f(u) = \sqrt{w(u, v)}\,(f(v) - f(u)). \tag{9}$$

This definition is consistent with the continuous definition of the directional derivative and has the following properties $\partial_v f(u) = -\partial_u f(v)$, $\partial_u f(u) = 0$, and if $f(u) = f(v)$ then $\partial_v f(u) = 0$.

Based on the definition of weighted finite differences in (9) one can straightforwardly introduce the *(nonlocal) weighted gradient* on graphs $\nabla_w : \mathcal{H}(V) \to \mathcal{H}(V \times V)$, which is defined on a vertex $u \in V$ as the vector of all weighted finite differences with respect to the set of vertices V, i.e.,

$$(\nabla_w f)(u) \;=\; (\partial_v f(u))_{v\in V}\,. \tag{10}$$

From the properties of the weighted finite differences above it gets clear that the weighted gradient is linear and antisymmetric. The weighted gradient in a vertex $u \in V$ can be interpreted as function in $\mathcal{H}(V)$ and hence the $\mathcal{L}^p(V)$ and $\mathcal{L}^\infty(V)$ norm in (7) of this finite vector represent its respective *local variation* and are given as:

$$\begin{aligned}\|(\nabla_w f)(u)\|_p \;&=\; \left(\sum_{v\sim u}(w(u,v))^{p/2}\,|f(v)-f(u)|^p\right)^{\frac{1}{p}},\\ \|(\nabla_w f)(u)\|_\infty &= \max_{v\sim u}\left(\sqrt{w(u,v)}\,|f(v)-f(u)|\right).\end{aligned} \tag{11}$$

Based on previous definitions, we can define two *upwind directional derivatives* expressed by :

$$\partial_v^{\pm} f(u) = \sqrt{w(u,v)}\big(f(v)-f(u)\big)^{\pm}, \tag{12}$$

with the notation $(x)^+ = \max(0,x)$ and $(x)^- = \max(0,-x)$.

Similarly, *discrete upwind nonlocal weighted gradients* are defined as

$$(\nabla_w^{\pm} f)(u) \overset{def.}{=} \left(\partial_v^{\pm} f(u)\right)_{v\in V}. \tag{13}$$

The *upwind gradient norm operators*, with $1 \leqslant p < \infty$ are defined for a function $f \in \mathcal{H}(V)$ as

$$\|(\nabla_w^{\pm} f)(u)\|_p = \left[\sum_{v\sim u}\sqrt{w(u,v)}^{\,p}\big(f(v)-f(u)\big)^{p\pm}\right]^{\frac{1}{p}}. \tag{14}$$

These operators allow to define the notion of the regularity of the function around a vertex u.

Similarly, in the case where $p=\infty$, *upwind gradient norm operators* are defined for a function $f \in \mathcal{H}(V)$ as

$$\|(\nabla_w^{\pm} f)(u)\|_\infty = \max_{v\sim u}\left(\sqrt{w(u,v)}\big(f(v)-f(u)\big)^{\pm}\right). \tag{15}$$

The relation between discrete gradient and this family of upwind gradients is given, for a function $f \in \mathcal{H}(V)$, by

$$\|(\nabla_w f)(u)\|_p^p = \|(\nabla_w^+ f)(u)\|_p^p + \|(\nabla_w^- f)(u)\|_p^p, \tag{16}$$

and one can deduce that

$$\|(\nabla_w^{\pm} f)(u)\|_p \leqslant \|(\nabla_w f)(u)\|_p. \tag{17}$$

Thus this family provides a slightly finer expression of the gradient. For instance, one can remark that $\|(\nabla_w^- f)(u)\|_p$ is always zero if f has a local minimum at u. The upwind discrete gradients was used in [8,27] to adapt the Eikonal Equation on weighted graphs, and to study the well-posedness (existence and uniqueness) of the solution with applications in image processing and Machine learning.

2.3 ∞-Laplacian on graph

The nonlocal ∞-Laplacian of a function $f \in \mathcal{H}(V)$, noted $\Delta_{w,\infty} : \mathcal{H}(V) \to \mathcal{H}(V)$ is defined by [1]

$$\Delta_{w,\infty} f(u) \overset{def.}{=} \frac{1}{2}\left[\|(\nabla_w^+ f)(u)\|_\infty - \|(\nabla_w^- f)(u)\|_\infty\right], \tag{18}$$

which can be rewritten as

$$\begin{aligned}\Delta_{w,\infty} f(u) = \frac{1}{2}\Big[&\max\Big(\sqrt{w(u,v)}\big(f(v)-f(u)\big)^+\Big)\\ &-\max\Big(\sqrt{w(u,v)}\big(f(v)-f(u)\big)^-\Big)\Big].\end{aligned} \tag{19}$$

Remark As in the continuous case, this operator can be formally derived as minimization of the following energy on graphs, as p goes to the infinity.

$$J_{w,p}(f) = \sum_{u\in V}\|(\nabla_w f)(u)\|_p^p. \tag{20}$$

For more details, see [2,9].

3 Nonlocal ∞-Poisson equation with gradient term

3.1 From tug-of-war game to PdEs on graphs

Let us rewrite the value function in the context of our PdE framework, considering the following Euclidean ε-adjacency graph $G = (V, E, w)$ with $V = \Omega \subset \mathbb{R}^n$, $E = \{(x,y) \in \Omega \times \Omega | w_0(x,y) > 0\}$ and

$$w_0(x,y) = \begin{cases}\frac{1}{\varepsilon^4}, & \text{if } |y-x| \leq \varepsilon,\\ 0, & \text{otherwise.}\end{cases}$$

By using w_0 in the discrete upwind gradient $\mathcal{L}_\infty$-norm, we get:

$$\begin{aligned}\|(\nabla^+_{w_0} f)(x)\|_\infty &= \max_{y\in B_\varepsilon}(\sqrt{w_0(x,y)}(f(y)-f(x)))\\ &= \frac{1}{\varepsilon^2}(\max_{y\in B_\varepsilon}(f(y))-f(x)),\end{aligned}$$

applying the same simplification to $\|(\nabla^-_{w_0} f)(x)\|_\infty$, we get:

$$\|(\nabla^-_{w_0} f)(x)\|_\infty = \frac{1}{\varepsilon^2}(f(x)-\min_{y\in B_\varepsilon}(f(y))).$$

We can define the sup and inf operator as:

$$\sup_{y\in B_\varepsilon(x)} f(y) = \varepsilon^2\|(\nabla^+_{w_0} f)(x)\|_\infty + f(x),$$

and

$$\inf_{y\in B_\varepsilon(x)} f(y) = f(x) - \varepsilon^2\|(\nabla^-_{w_0} f)(x)\|_\infty.$$

Now, by replacing them in Eq. (4), we get:

$$\begin{aligned}f(x) =& \frac{\varepsilon^2}{2}\left[\|(\nabla^+_{w_0} f)(x)\|_\infty - \|(\nabla^-_{w_0} f)(x)\|_\infty\right] + f(x)\\ &+ \varepsilon^2 h(x),\end{aligned}$$

which can be simplified as:

$$\Delta_{w_0,\infty} f(x) = -h(x), \tag{21}$$

which is the discrete ∞-Poisson equation. Now if we use a general weight function, we get a general discrete ∞-Poisson equation on graph:

$$\Delta_{w,\infty} f(u) = -h(u),$$

Similarly, using our discrete PdE framework, we can transcribe the biased tug-of-war (Eq. (6)) game as:

$$\alpha\|(\nabla^+_w f)(x)\|_\infty - \beta\|(\nabla^-_w f)(x)\|_\infty + h(x) = 0 \tag{22}$$

with $\alpha, \beta \in [0,1]$, and $\alpha+\beta = 1$.

We define the operator $\mathcal{L}_{w,\infty} f(u)$ as:

$$\mathcal{L}_{w,\infty} f(u) \stackrel{def}{=} \alpha\|(\nabla^+_w f)(u)\|_\infty - \beta\|(\nabla^-_w f)(u)\|_\infty. \tag{23}$$

It corresponds to a new family of ∞-Laplace operators with gradient terms.

By a simple factorization, this operator can be rewritten as

$$\begin{aligned}\mathcal{L}_{w,\infty} f(u) =& 2\min(\alpha,\beta)\Delta_{w,\infty} f(u)\\ &+ (\alpha-\beta)^+\|(\nabla^+_w f)(u)\|_\infty\\ &- (\alpha-\beta)^-\|(\nabla^-_w f)(u)\|_\infty.\end{aligned} \tag{24}$$

We consider the following equation that describes the general Dirichlet problem associated to the Poisson equation on graphs:

$$\begin{cases}-\mathcal{L}_{w,\infty} f(u) = h(u) & u \in A\\ f(u) = g(u) & u \in \partial A,\end{cases} \tag{25}$$

where A is a connected set of vertices and ∂A its boundary.

One can see that using different values for α and β in $\mathcal{L}_{w,\infty} f(u)$, we can recover different version of the equation. In this work, we are particularly interested by three of them:

– case $\alpha = \beta \neq 0$, expression of (24) becomes

$$\mathcal{L}_{w,\infty} f(u) = \Delta_{w,\infty} f(u), \tag{26}$$

and recovers the discrete ∞-Laplacian expressions. Eq. (25) now becomes:

$$-\Delta_{w,\infty} f(u) = h(u) \tag{27}$$

– case $\alpha - \beta < 0$, expression of (24) becomes

$$\begin{aligned}\mathcal{L}_{w,\infty} f(u) =& 2\alpha\Delta_{w,\infty} f(u)\\ &- (\beta-\alpha)\|(\nabla^-_w f)(u)\|_\infty.\end{aligned} \tag{28}$$

Eq. (25) now becomes:

$$-2\alpha\Delta_{w,\infty} f(u) + (\beta-\alpha)\|(\nabla^-_w f)(u)\|_\infty = h(u) \tag{29}$$

– case $\beta = 1$, it becomes

$$\mathcal{L}_{w,\infty} f(u) = -\|(\nabla^-_w f)(u)\|_\infty. \tag{30}$$

We can see here that we recover PdEs based morphological operators with the upwind derivative discretization. Eq. (25) now becomes:

$$\|(\nabla^-_w f)(u)\|_\infty = h(u), \tag{31}$$

By setting $h(u) = -1$ and $g(u) = 0$, this formulation recovers the following eikonal equation:

$$\begin{cases}\|(\nabla^-_w f)(u)\|_\infty = 1, & u \in A\\ f(u) = 0, & u \in \partial A.\end{cases} \tag{32}$$

By setting different interaction functions (defining the weight of the graph), we can compute a generalized distance, from any vertex in A to ∂A. This

eikonal equation is a particular case of a more generalized equation on weighted graphs:

$$\begin{cases} \|(\nabla_w^- f)(u)\|_p = 1, & u \in A \\ f(u) = 0, & u \in \partial A. \end{cases} \tag{33}$$

This family of equation has been studied in [8]. In particular, for $p = 2$, it has been shown that when dealing with grid graph, this equation corresponds to the Osher-Sethian discretization scheme [19].

In the next Sections, to solve the hybrid ∞-Poisson equation, we set $h = -1$, $g = 0$, and $\alpha \leq \beta$.

3.2 Connection with nonlocal game and PDEs

In this subsection, we show that Eq.(21) is related to the nonlocal version of the tug-of-war game and to the continuous version of the nonlocal Hölder ∞-Laplacian. Considering the same game, but replacing the ε-ball by a neighborhood $N(x_{k-1}) \subset \Omega$ where:

$$N(x_{k-1}) = \{x \in \Omega | w(x, x_{k-1}) > 0\}. \tag{34}$$

In this version of the game, the game token is then moved to x_k, where x_k is chosen randomly so that $x_k = x_k^I$ with a probability

$$P = \frac{\sqrt{w(x_{k-1}, x_k^I)}}{\sqrt{w(x_{k-1}, x_k^I)} + \sqrt{w(x_{k-1}, x_k^{II})}}, \tag{35}$$

and that $x_k = x_k^{II}$ with a probability $1 - P$. According to the dynamic programming principle, the value functions for Player I and Player II for this game satisfy the relation

$$\begin{aligned} &\frac{1}{2}\big[\max_{y \in N(x)} \sqrt{w(x,y)}\big(f(y) - f(x)\big) + \\ &\min_{y \in N(x)} \sqrt{w(x,y)}\big(f(y) - f(x)\big)\big] = -h(x), \end{aligned} \tag{36}$$

which is simply

$$\Delta_{\infty,w} f(x) = -h(x). \tag{37}$$

Now, if we consider a nonlocal Euclidean graph $G = (V, E, w)$ with $V = \Omega \subset \mathbb{R}^n$, $E = \{(x, y) \in \Omega \times \Omega | w_1(x, y) > 0\}$, and the following weight function :

$$w_1(x,y) = \begin{cases} \frac{1}{|x-y|^{2s}}, & \text{if } x \neq y, s \in [0,1], \\ 0, & \text{otherwise.} \end{cases} \tag{38}$$

Our formulation of $\Delta_{w,\infty}$ corresponds to the recently proposed Hölder ∞-Laplacian proposed by Chambolle et al. in [5].

$$\begin{aligned} \Delta_{w_1,\infty} f(x) = \frac{1}{2}\Bigg[&\max_{y \in \Omega, y \neq x} \left(\frac{f(y) - f(x)}{|y - x|^s}\right) \\ &+ \min_{y \in \Omega, y \neq x} \left(\frac{f(y) - f(x)}{|y - x|^s}\right)\Bigg]. \end{aligned} \tag{39}$$

This operator is formally derived from the minimization of an energy of the form

$$\int_\Omega \int_\Omega \frac{[f(y) - f(x)|^p}{|x - y|^{p \times s}} dx dy, \tag{40}$$

as $p \to \infty$.

Now, considering the biased tug-of-war, we get the probability to get $x_k = x_k^I$ as:

$$P = \frac{\alpha\sqrt{w(x_{k-1}, x_k^I)}}{\alpha\sqrt{w(x_{k-1}, x_k^I)} + \beta\sqrt{w(x_{k-1}, x_k^{II})}}. \tag{41}$$

Following the same reasoning, we get the relation:

$$\mathcal{L}_{w_1,\infty} f(u) = -h(u). \tag{42}$$

The formulation of the operator $\mathcal{L}_{w_1,\infty} f(u)$ also corresponds to the Hölder infinity Laplacian, but with gradient terms :

$$\begin{aligned} \mathcal{L}_{w_1,\infty} f(u) = &2\min(\alpha, \beta)\Bigg[\max_{y \in \Omega, y \neq x} \left(\frac{f(y) - f(x)}{|y - x|^s}\right) \\ &+ \min_{y \in \Omega, y \neq x} \left(\frac{f(y) - f(x)}{|y - x|^s}\right)\Bigg] \\ &+ (\alpha - \beta)^+ \max_{y \in \Omega, y \neq x} \left(\frac{f(y) - f(x)}{|y - x|^s}\right) \\ &- (\alpha - \beta)^- \min_{y \in \Omega, y \neq x} \left(\frac{f(y) - f(x)}{|y - x|^s}\right). \end{aligned} \tag{43}$$

3.3 Generalized distance computation on graph

Like the case of random walk for $\Delta_2 f(u)$, we are interested here to the case where $h(u) = 1$ and $g(u) = 0$ in Eq. (25), leading to the following equation:

$$\begin{cases} \mathcal{L}_{w,\infty} f(u) = -1 & u \in A \\ f(u) = 0 & u \in \partial A. \end{cases} \tag{44}$$

To solve this equation, we first simplify it as $\mathcal{L}_{w,\infty} f(u) + 1 = 0$ in order to set the dynamic following scheme:

$$\begin{cases} \frac{\partial f(u,t)}{\partial t} = \mathcal{L}_{w,\infty} f(u,t) + 1 & u \in A \\ f(u,t) = 0 & u \in \partial A \\ f(u,t=0) = 0 & u \in A. \end{cases} \tag{45}$$

To discretize the time variable in Eq. (45), we use the explicit Euler time discretization method: $\frac{\partial f(u,t)}{\partial t} = \frac{f^{n+1}(u) - f^n(u)}{\Delta t}$, where $f^n(u) = f(u, n\Delta t)$, in order to get the following iterative scheme:

$$\begin{cases} f^{n+1}(u) = f^n(u) + \Delta t(\mathcal{L}_{w,\infty} f(u,t) + 1) & u \in A \\ f^{n+1}(u) = 0 & u \in \partial A \\ f^0(u) = 0 & u \in A. \end{cases} \tag{46}$$

This iterative scheme can be interpreted as a morphological scheme implying morphological type filter. We define the following operators:

$$\begin{aligned} \mathrm{NLD}(f)(u) &= \|(\nabla_w^+ f)(u)\|_\infty + f(u), \\ \mathrm{NLE}(f)(u) &= f(u) - \|(\nabla_w^- f)(u)\|_\infty. \end{aligned} \tag{47}$$

NLD and NLE refers to nonlocal Dilation, and nonlocal Erosion, respectively. The reason we call them this way is that they correspond to classical erosion and dilation on weighted graphs. By setting $\Delta t = 1$ and rewritting the iterative scheme (46) using (47), we get the following iterative algorithm:

$$\begin{cases} f^{n+1}(u) = \alpha \mathrm{NLD}(f)(u) + \beta \mathrm{NLE}(f)(u) + 1 & u \in A \\ f^{n+1}(u) = 0 & u \in \partial A \\ f^0(x) = 0 & u \in A. \end{cases} \tag{48}$$

4 Applications

4.1 Graph construction

There exists several popular methods to transform discrete data $\{x_1, ...x_n\}$ into a weighted graph structure. Considering a set of vertices V such that data are embedded by functions of $\mathcal{H}(V)$, the construction of such graph consists in modeling the neighborhood relationships between the data through the definition of a set of edges E and using a pairwise distance measure $\mu : V \times V \to \mathbb{R}^+$. In the particular case of images, the ones based on geometric neighborhoods are particularly well-adapted to represent the geometry of the space, as well as the geometry of the function defined on that space. One can quote:

- *Grid graphs* which are most natural structures to describe an image with a graph. Each pixel is connected by an edge to its adjacent pixels. Classical grid graphs are 4-adjacency grid graphs and 8-adjacency grid graphs. Larger adjacency can be used to obtain nonlocal graphs.
- *Region adjacency graphs* (RAG) which provide very useful and common ways of describing the structure of a picture: vertices represent regions and edges represent region adjacency relationship.
- *k-neighborhood graphs* (k-NNG) where each vertex v_i is connected with its k-nearest neighbors according to μ. Such construction implies to build a directed graph, as the neighborhood relationship is not symmetric. Nevertheless, an undirected graph can be obtained while adding an edge between two vertices v_i and v_j if v_i is among the k-nearest neighbor of v_j or if v_j is among the k-nearest neighbor of v_i
- *k-Extended RAG* (k-ERAG) which are RAGs extended by a k-NNG. Each vertex is connected to adjacent regions vertices and to it's k most similar vertices of V.

The similarity between two vertices is computed according to a measure of similarity $g : E \to \mathbb{R}^+$, which satisfies:

$$w(u,v) = \begin{cases} g(u,v) & \text{if } (u,v) \in E \\ 0 & \text{otherwise} \end{cases}$$

Usual similarity functions are as follow:

$$\begin{aligned} g_0(u,v) &= 1, \\ g_1(u,v) &= exp\Big(-\mu\big(f^0(u), f^0(v)\big)/\sigma^2\Big) \text{ with } \sigma > 0, \\ g_2(u,v) &= \frac{1}{\mu\big(f^0(u), f^0(v)\big)}, \end{aligned}$$

where σ depends on the variation of the function μ and control the similarity scale.

Several choices can be considered for the expression of the feature vectors, depending on the nature of the features to be used for the graph processing. In the context of image processing, one can quote the simplest gray scale or color feature vector F_u, or the patch feature vector $F_u^\tau = \bigcup_{v \in \mathcal{W}^\tau(u)} F_v$ (i.e, the set of values F_v where v is in a square window $\mathcal{W}^\tau(u)$ of size $(2\tau+1) \times (2\tau+1)$ centered at a vertex pixel u), in order to incorporate nonlocal features.

4.2 Weighted geodesic distances

4.2.1 Synthetic image

To illustrate the effect of solving our proposed adaptation of the ∞-Poisson equation, we experimented our algorithm on a synthetic image (first image of Fig. 1). The results were obtained using an 8-adjacency graph, using two different weight functions, and different α, β values. The original image is a 400×400 grayscale image, and the distance is computed from the top left corner of the image. The first row shows distance maps with isolines obtained using the weight function g_0 ($w(u,v) = 1$). As one can see, using $\alpha = 0$, the distance between two adjacent isolines is constant, depicting a linear distance function. One can also observe that the computed distance is anisotropic, giving more importance to the diagonal directions (this is fully expected and due to the $\mathcal{L}^\infty$-norm). As α varies from 0 to 0.5, the computed distance is evolving from a linear to a quadratic one. The second row illustrates the effects of the weight function g_1, where the distance function μ we used here is the $\mathcal{L}^2$ distance between the pixel intensities, with $\sigma = 150$. The shape information is naturally represented by this weight function in the graph, enhancing the value of the computed distance as it reaches the boundary of an object.

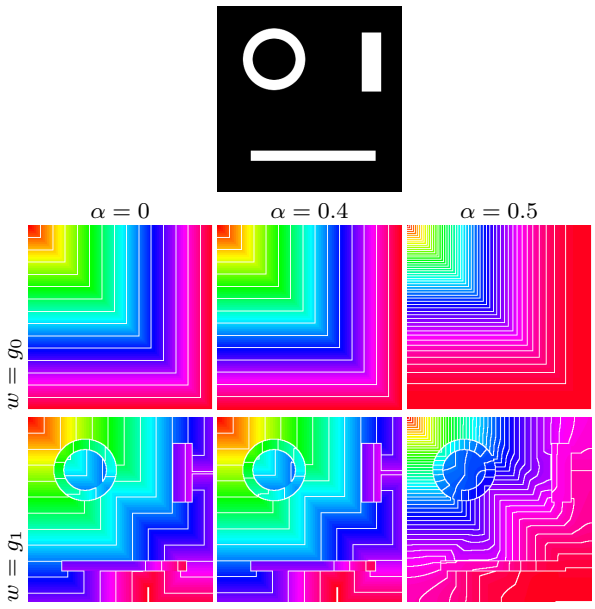


Fig. 1 Distance computation on a synthetic image, with different α values, and different weight functions. The distance is computed from the top left corner of the image.

4.2.2 Shapes

We also experimented our algorithm on shapes images (Fig. 2). The graph is built the same way as for the experiments on the synthetic image, and we used $w(u,v) = g_2(u,v)$, with μ the $\mathcal{L}^2$ distance function between pixels coordinate. In this case, the distance is computed from the boundaries of the shape. As one can see, by varying the α parameter, we can observe the same phenomena as for the synthetic image, namely the distance function is evolving from linear to quadratic.

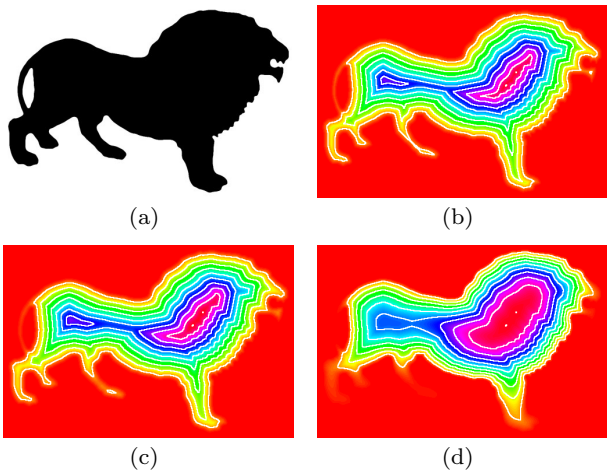


Fig. 2 Distance generation on a shape image using different α values: Fig. (b) is computed with $\alpha = 0$, Fig. (c) with $\alpha = 0.4$, and Fig. (d) with $\alpha = 0.5$. The distance is computed from the boundary of the shape.

4.2.3 Natural image

To show the effects of the weight function, we experimented our algorithm on a natural image (top image of Fig. 3). On the first, second, and third rows, we also used $w(u,v) = g_2(u,v)$, with μ the $\mathcal{L}^2$ distance function between pixel coordinates. The first row shows the computed distance with a 4-adjacency grid graph, second row with an 8-adjacency one, and third row with a 7×7 square neighborhood window (*i.e.* a node u is connected to all the point in a 7×7 square window, centered at node u). As one can see, by adding more neighbors to a node, the computed distance tends to be less anisotropic. In fact, it is still anisotropic, but in the direction of the neighboors. As we added more neighbors to the graph, there is more and more directions to take into account. To get an isotropic distance, a solution would be to add an infinity of neighboors in an infinity of direction. The fourth row shows the distance computed by building an 8-adjacency graph, $g_1(u,v)$ as

a color similarity weight function. As for the synthetic image, one can see that this weight function permits to exhibit shapes according to the pixel intensity differences. On the fifth row, we built a k-nn graph, in the patch space of the image. We chose $k = 5$, the research being performed in a 15×15 pixels window around the pixel. We used patches of size 5×5 pixels, also centered at the corresponding pixel. To compute the similarity, we used the g_1 function, with μ the normalized sum of the $\mathcal{L}^2$ distance between each pixels of the patch of the considered pair of nodes. As one can see, the objects are even better delineated, showing large areas of slow evolution of the distance when in a same object, and fast evolution at the edge of an object.

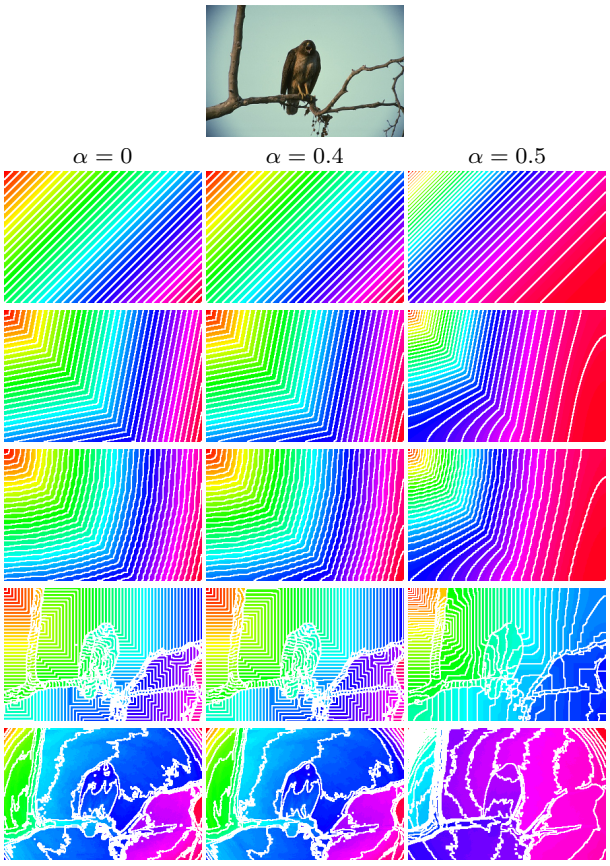


Fig. 3 Distance computation on a natural image, using different α values, different graph construction, and different weight functions. First row is generated with a 4-adjacency grid graph, second row with an 8-adjacency grid graph, and third row with neighbors in a 7×7 window around the pixel. The weight function for these latter is the Euclidean distance in the coordinate space of the pixel's grid. For the fourth row, we used an 8-adjacency grid graph, with color similarity as a weight function. The fifth row has been generated using a $k$nn graph in the patch space of the image. See text for more details.

4.2.4 3D point cloud

To show the adaptivity of our framework, we computed the generalized distance on several point clouds (Fig. 4). We built the graph as a k-nn with $k = 5$, in the coordinate space of the point cloud. As the spatial discretization step is regular enough, we used a constant weight function ($w(u, v) = 1$). The superimposed red line on the figure is the shortest path between the source point (the point from which the distance is computed) and an other point in the point cloud. This path was obviously computed using the computed distance function. We compared ourselves with the adaptation of the eikonal equation on graph [8] (first column of Fig. 4) using the $\mathcal{L}^2$ norm of the gradient.

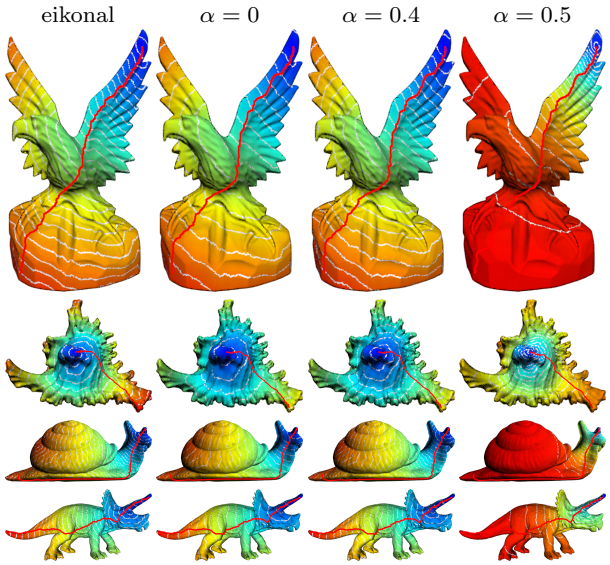


Fig. 4 Distance generation on 3D point clouds data, using different values of α and a $k$nn graph. See text for more details.

4.3 Semi-supervised segmentation and data clustering

In this section, we present the behavior of our algorithm for the task of semi-supervised image segmentation and semi-supervised data classification. We illustrate it with local and nonlocal configuration on different kind of data, through several examples.

In the case of image segmentation, several approaches have become very popular, such as graph cuts [3], random walk [12], shortest-path, watershed or framework that unify some of the previous methods (as powerwatershed) [6,25].

4.3.1 Label diffusion algorithm

The presented algorithm to compute generalized distances on weighted graphs may also be used for the task of image semi-supervised segmentation. This task can be seen as a label diffusion one. To accomodate our algorithm to the label diffusion, we rewrite it as follows: Let $V = \{u_1, ..., u_n\}$ be a finite set of data, where each data u_i is a vector of $\mathbb{R}^m$, and let $G = (V, E, w)$ be a weighted graph such that data points are vertices and are connected by an edge of E. The semi supervised segmentation of V consists in partitioning the set V into k classes (known beforehand) given inital labels for some vertices of V. The aim is then to estimate the unlabeled data from the labeled ones. Let C_l be a set of labeled vertices, these latter belonging to the l^{th} class. Let $V_0 = \bigcup\{C_l\}_{l=1,...,k}$ be the set of initial *labeled* vertices and let $V \backslash V_0$ be the initial *unlabeled* vertices. Then, the vertex labeling is performed by k independent distance computation from its corresponding set of initial label vertices:

$$\begin{cases} \mathcal{L}_{w,\infty} f_l(u) = -1 & u \in V \backslash V_0 \\ f_l(u) = 0 & u \in C_l. \end{cases} \tag{49}$$

At the end of the distance computation, the class membership of a node u is given as the label of the smallest distance function: $\arg\min_{l \in 1,...,k} f_l(u)$.

Fig. 5 Image segmentation using local and nonlocal graph construction. See text for details.

4.3.2 Image segmentation

We illustrated this method on images in the Fig. 5. The first image is the initial image with superimposed initial labels. For the second image, we built an 8-adjacency grid graph, with a weight function $w(u, v)$ the color similarity. For the third image, we built a nonlocal k-nn graph, in patch space of the image the same way as we did for the natural image distance computation illustration (Fig. 3). We also illustrated the effects of using the nonlocal configuration of the graph using patch

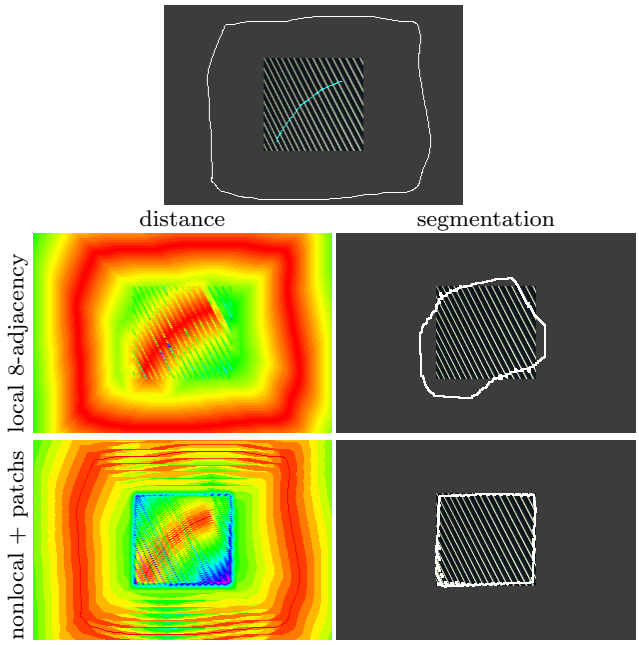


Fig. 6 Segmentation of a texture image. First image is the initial image with superimposed initial labels. First row is computed using an 8-adjacency grid graph with color similarity. Second row is computed using a k-nn graph with $k = 15$ in the patch space of the image. See text for more details.

similarity on a texture image (Fig. 6). As for Fig. 5, the first image is the initial image with superimposed inital labels. In the left column, the computed distance is shown, and on the right the final segmentation. First line shows the local results, also using an 8-adjacency grid graph. The last row presents nonlocal results with a 15×15 window and patchs of 9×9. These results show the benefits of non-local configurations using patches, especially for textured images, where classical methods fail to find correctly the desired object.

To provide a quantitative assessment of the proposed method, we use the Microsoft Grabcut database [22], which is available online. We borrowed the results of the experiments made by the authors of [6], where they compare themselves with the previously cited methods : graph cuts [3], random walk [12], shortest-path, and maximum spanning forest (MSF). We evaluated our algorithm by quantifying the errors of the results segmentation using some of the measure used in [6], *i.e.* Boundary Error (BE), Rand Index (RI), and Global Consistency Error (GCE). The Boundary Error between two segmented images measures the average distance of a boundary pixel in the first image and its closest in the second image. The Rand Index counts the fraction of pairs of pixels whose labels are consistent between the computed segmentation and the ground truth. It takes values in the range $[0, 1]$. The Global Consistency Error measures the extent to which one segmentation can

Table 1 Grabcut assessment

	BE	RI	GCE
Shortest paths	2.82	0.972	0.0233
Random walker	2.96	0.971	0.0234
MSF	2.89	0.971	0.0244
Power wshed	2.87	0.971	0.0245
Graph cuts	3.12	0.970	0.0249
∞-Poisson ($\alpha = 0$)	1.25	0.977	0.0198
∞-Poisson ($\alpha = 0.4$)	1.23	0.977	0.0198
∞-Poisson ($\alpha = 0.48$)	1.21	0.977	0.02

be viewed as a refinment of the other. A good segmentation is characterised by a BE and a GCE as small as possible, and a RI as close to 1 as possible. For this experiment, we built an 8-adjacency grid graph, with $w(u, v) = g_1(u, v)$. The results of this experiment are shown in table 1. As one can see, our algorithm is slightly better through all the error measurments, denoting better segmentation on this dataset.

4.3.3 Data classification

We also experimented our method to perform data clustering on samples picked from the MNIST database [16] (Fig. 7), to show the adaptivity and behavior of the proposed algorithm for high dimensional unorganised data clustering. The method is similar to the image segmentation case, and can be adapted straightforwardly: to build the graph, each node represent an object of the database. For this case, we represented each object by its corresponding pixels vector. To compute the similarity between objects, we used the g_1 weight function, with μ the $\mathcal{L}^2$ distance between each pixel vectors. We then built a k-nn graph in this same space, that is represented in Fig. 7 (a). To achieve the clustering, two vertices of each class were chosen randomly as initial labels. Fig. 7 (b) shows the final clustering.

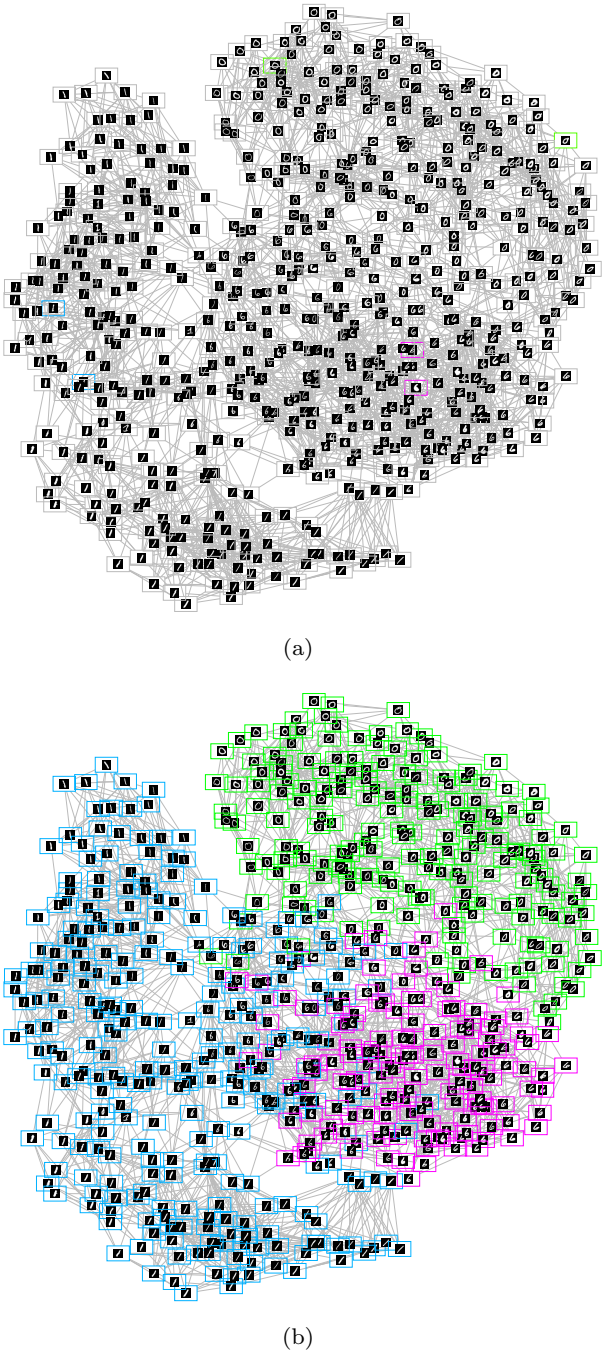


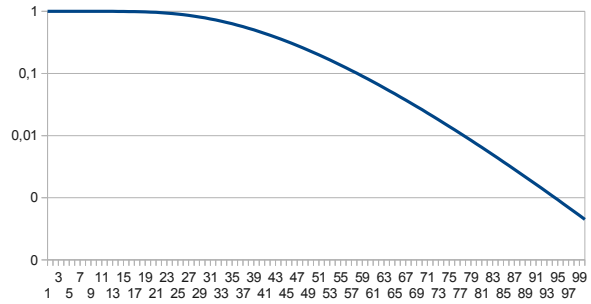
Fig. 7 MNIST sample data clustering. The graph is a k-nn graph. Images are represented as a vector of pixels and similarity is computed using the $\mathcal{L}^2$ distance between these vectors. Fig. (a) represents the graph with initial labels, while Fig. (b) represents the semi-supervised classification using these initials labels.

4.4 Computational efficiency

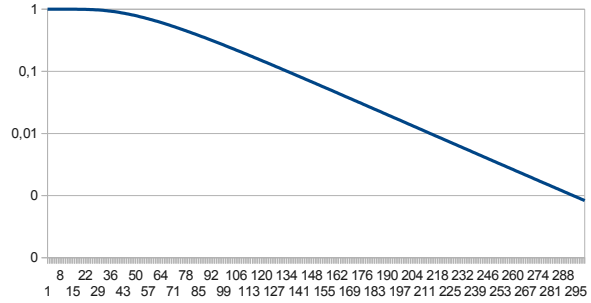
In this paragraph we discuss about the computational efficiency and scalability of the algorithm. As the main goal of this paper is to present an adaptation of the ∞-Poisson on graphs, in order to provide a way to compute distances on graphs of arbitrary topology, we did not focus on the optimal way to solve the equation and used a simple time discretization method to get our iterative algorithm. In a nutshell, by changing the parameter α, the convergence rate is superlinear using $\alpha = 0$ (Fig. 8(a)), and sublinear using $\alpha = 0.5$(Fig. 8(c)).

5 Conclusion

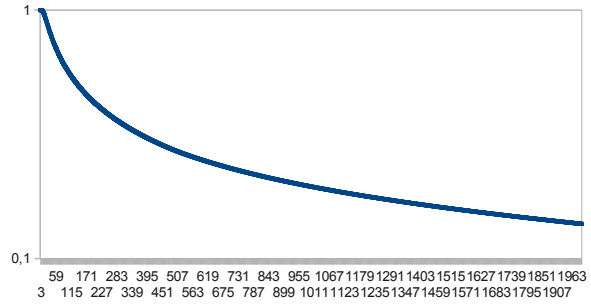
In this paper, by showing the link between the tug-of-war game and the ∞-Poisson equation, we were able to



(a)



(b)



(c)

Fig. 8 Convergence rate of the proposed algorithm with different values of α ($\alpha = 0$ in Fig. (a), 0.25 in Fig. (b), and 0.5 in Fig. (c)).

adapt its formulation on graph, and propose a finer expression of the ∞-Laplace operator with gradient term, by making the link with the biased version of the tug-of-war game. We used this fomulation to adapt and extend the ∞-Poisson equation on graph, exhibiting some special cases of the equation by making vary the gradient's coefficient, showing the link between this version of the ∞-Poisson equation and the eikonal equation. We used this extension on graph to compute distances on data that can be represented as a graph: images, 3D point cloud, unorganised n-dimensional data. We have also shown that this formulation can be used to compute image segmentation and data clustering.

Acknowledgements This work was supported under a doctoral grant of the Conseil Régional de Basse Normandie and of the Coeur et Cancer association in collaboration with the Department of Anatomical and Cytological Pathology from Cotentin Hospital Center, and by a european FEDER grant (OLOCYG project).

References

1. Abderrahim, E., Xavier, D., Zakaria, L., Olivier, L.: Non-local infinity laplacian equation on graphs with applications in image processing and machine learning. Mathematics and Computers in Simulation **102**, 153–163 (2014)
2. Bougleux, S., Elmoataz, A., Melkemi, M.: Local and non-local discrete regularization on weighted graphs for image and mesh processing. International Journal of Computer Vision **84**(2), 220–236 (2009)
3. Boykov, Y.Y., Jolly, M.P.: Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In: Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on, vol. 1, pp. 105–112. IEEE (2001)
4. Calakli, F., Taubin, G.: Ssd: Smooth signed distance surface reconstruction. In: Computer Graphics Forum, vol. 30, pp. 1993–2002. Wiley Online Library (2011)
5. Chambolle, A., Lindgren, E., Monneau, R.: A hölder infinity laplacian (2011)
6. Couprie, C., Grady, L., Najman, L., Talbot, H.: Power watershed: A unifying graph-based optimization framework. Pattern Analysis and Machine Intelligence, IEEE Transactions on **33**(7), 1384–1399 (2011)
7. Crane, K., Weischedel, C., Wardetzky, M.: Geodesics in heat: a new approach to computing distance based on heat flow. ACM Transactions on Graphics (TOG) **32**(5), 152 (2013)
8. Desquesnes, X., Elmoataz, A., Lézoray, O.: Eikonal equation adaptation on weighted graphs: fast geometric diffusion process for local and non-local image and data processing. Journal of Mathematical Imaging and Vision **46**, 238–257 (2013)
9. Elmoataz, A., Desquesnes, X., Lézoray, O.: Non-local morphological pdes and-laplacian equation on graphs with applications in image processing and machine learning. Selected Topics in Signal Processing, IEEE Journal of **6**(7), 764–779 (2012)
10. Elmoataz, A., Lezoray, O., Bougleux, S., Ta, V.: Unifying local and nonlocal processing with partial difference operators on weighted graphs. In: International Workshop on Local and Non-Local Approximation in Image Processing (LNLA), pp. 11–26 (2008)
11. Gorelick, L., Galun, M., Sharon, E., Basri, R., Brandt, A.: Shape representation and classification using the poisson equation. Pattern Analysis and Machine Intelligence, IEEE Transactions on **28**(12), 1991–2005 (2006)
12. Grady, L.: Random walks for image segmentation. Pattern Analysis and Machine Intelligence, IEEE Transactions on **28**(11), 1768–1783 (2006)
13. Jeong, W.K., Whitaker, R.T.: A fast iterative method for eikonal equations. SIAM Journal on Scientific Computing **30**(5), 2512–2534 (2008)

14. Kawohl, B.: On a family of torsional creep problems. J. reine angew. Math **410**(1), 1–22 (1990)
15. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Proceedings of the fourth Eurographics symposium on Geometry processing (2006)
16. LeCun, Y., Cortes, C.: MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/ (2010). URL http://yann.lecun.com/exdb/mnist/
17. Mémoli, F., Sapiro, G.: Fast computation of weighted distance functions and geodesics on implicit hyper-surfaces. Journal of computational Physics **173**(2), 730–764 (2001)
18. Oberman, A.M.: Finite difference methods for the infinity laplace and p-laplace equations. Journal of Computational and Applied Mathematics **254**, 65–80 (2013)
19. Osher, S., Sethian, J.: Fronts Propagating With Curvature-dependent Speed - Algorithms Based On Hamilton-Jacobi Formulations. Journal Of Computational Physics **79**(1), 12–49 (1988)
20. Peres, Y., Pete, G., Somersille, S.: Biased tug-of-war, the biased infinity laplacian, and comparison with exponential cones. Calculus of Variations and Partial Differential Equations **38**(3-4), 541–564 (2010)
21. Peres, Y., Schramm, O., Sheffield, S., Wilson, D.: Tug-of-war and the infinity laplacian. Journal of the American Mathematical Society **22**(1), 167–210 (2009)
22. Rother, C., Kolmogorov, V., Blake, A.: Grabcut: Interactive foreground extraction using iterated graph cuts. ACM Transactions on Graphics (TOG) **23**(3), 309–314 (2004)
23. Ruuth, S.J., Merriman, B.: A simple embedding method for solving partial differential equations on surfaces. Journal of Computational Physics **227**(3), 1943–1961 (2008)
24. Sethi, M., Rangarajan, A., Gurumoorthy, K.: The schrödinger distance transform (sdt) for point-sets and curves. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pp. 198–205. IEEE (2012)
25. Sinop, A.K., Grady, L.: A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm. In: ICCV, pp. 1–8. Citeseer (2007)
26. Ta, V., Elmoataz, A., Lézoray, O.: Nonlocal pdes-based morphology on weighted graphs for image and data processing. IEEE transactions on Image Processing **20**(6), 1504–1516 (2011)
27. Ta, V.T., Elmoataz, A., Lézoray, O.: Adaptation of eikonal equation over weighted graph. In: Scale Space and Variational Methods in Computer Vision, pp. 187–199. Springer (2009)